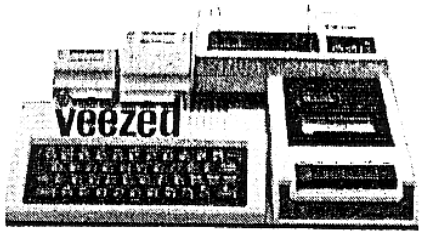
MICRO MAGIC



Beginner's Guide to the Veezed ~2

MICRO TYPE

MICRO TYPE is a BASIC Word Processor written for the VeeZeds 200 and 300. As presented here it is suitable for use with a Dick Smith X-3250 Graphic Printer, otherwise known as a GP-100. In this issue of MICRO MAGIC we give instructions on how to adapt the program for use with other Centronics-type printers, and for use with the Dick Smith PP40 Printer Plotter and the TOYO TP-40 Printer Plotter. Please read the operating instructions before entering the program so you can understand the commands.

MICRO TYPE Instructions

CRUN

- Select ENTER or RETRIEVE
- ENTER is for entering text.
- RETRIEVE is for loading text from tape.

ENTERING TEXT.

- A TYPING GUIDE will appear with a flashing curser against the first place of the first line of text to be typed. Using INVERSE for Capital letters, Commas, Colons and Semi-colons type a message. Check for correctness and hit RETURN to access the MENU. To move the curser along the line of text use CTRL and either the left or right facing ARROWS. The curser does not need to be at the end of the line of text when you hit RETURN.

MENU.

- The MENU appears after each line of text. The INVERSE capital letter of the options are the letters to press to select the options. To CONTINUE typing hit C and the next line becomes available with the previous line printed below as an aid to composing.

INDENT.

- Indent means starting a line a few spaces in from the left-hand margin. Often used in the first line of a paragraph. To INDENT type a SHIFTZ graphic into first place of a line and then SPACE to where you want the sentence to begin. The SHIFTZ graphic is a non-printing character.

SCROLL.

- To review all that you have typed, from the MENU select S (for SCROLL) and your text will be printed line-by-line preceded by the line number. Make a note of the number of any line needing correction.

EDIT.

- From the MENU select E (for EDIT) and enter the number of the line requiring correction. The line will appear on the screen with TYPING GUIDES and the word DELETE. You are required to type the word or words to be Deleted. After pressing the RETURN the word(s) to be deleted will be erased and the word changed to ENTER. Type in the word(s) that are to take the place of those being deleted. On pressing RETURN the line will be amended on-screen. It is important that the word(s) to be deleted are EXACTLY the same as was originally typed. The computer will not recognise any variation.

PRINT

- From the MENU select P for PRINT and your text will be printed on your Printer.

FILE

- Set up a cassette for recording. Select F for FILE and start tape on RECORD. Each line will be printed as it is recorded. Keep a note of the number of lines for future retrieval.

RETRIEVE

- Set up cassette ready to PLAY. Select R for RETRIEVE either from the initial option after CRUN or from the Menu. Then ENTER the number of lines to be retrieved. Enter RETURN and when WAITING appears start the tape. As each line is retrieved it will be printed on the screen.

BREAK

- A useful feature is that you can BREAK the program without losing your text, providing you do not use LLIST or LIST. If, for instance, you had forgotten how many lines there were to be retrieved. Enter a larger number than is likely. Begin to RETRIEVE. When the last line appears on the screen, stop the tape, BREAK the program and type GOTO230<RETURN> and the Menu will appear. The number of lines will be the number that you used. By using SCROLL you can determine how many lines there really is. Again, BREAK the program and type L=25:GOTO230<RETURN> (L being the actual number of lines) and the Menu will return and the number of lines will be correct.
- Also, if you wanted to DELETE a complete line, instead of using EDIT and re-typing the complete line to be DELETED, you could BREAK and if, say, you wanted to DELETE the whole of line 25, type Z$(25)="■":GOTO230<RETURN> making the charcter inside the quotes a SHIFTZ Graphic. Then when the Menu reappears you only need to EDIT line 25. It is important to remember that no String Variable should ever be left empty. That is why we made Z$(25) a non-printing graphic, so that it can be EDITed if need be.

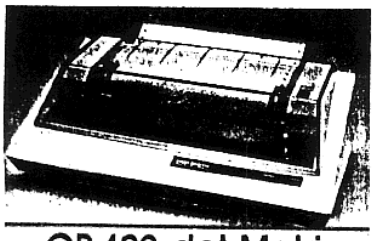
AUTO

- If you want to use AUTOmatic Line Numbering here it is:-

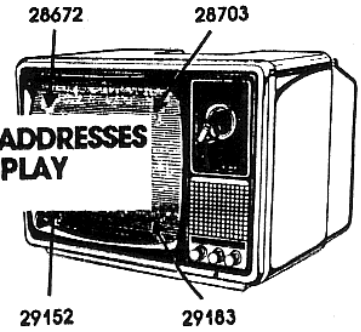
```
0  REM100,10
1  POKE31469,183:RUN
```

Note that RUN is part of line 1.
ENTER **RUN** and line number 100 will appear.

Pages 1 to 4 were prepared using a Dick Smith GP-100 Printer. This printer simulates the Graphics and Inverse characters of VeeZed computers. The rest of MICRO MAGIC was prepared on a STAR SG-10 Printer that does not print VeeZed Graphics or the Inverse characters. The advantages and disadvantages of these printers will be discussed after the MICRO TYPE program. How to adapt this program to suit different printers and plotters will also be given.



GP-100 dot Matrix Printer



DECIMAL POKE ADDRESSES FOR VZ-200 DISPLAY CORNERS

*** M I C R O   T Y P E ***

Be PreaPared to CSAVE as you Proceed.

```
100 CLS:CLEAR10000:DIMZ$(100)
```

Clear the screen. CLEAR 10000 will clear 10,000 memory locations while DIMZ$(100) will allow Z$(0) to Z$(100) to be used as String Variables for storing text.
■ If your VeeZed is unexPanded then use the following:-
100 CLS:CLEAR2000:DIMZ$(60)
It is recommended that you limit your lines of text to 50, keeping the extra 10 in reserve for finishing sentences.

```
110 POKE30744,0:PRINT@230,"■MICRO TYPE■":SOUND16,8:L=1:COLOR,1
```

After Printing the title with a SHIFTJ at either end of the INVERSE letters of MICRO TYPE, put in a SOUND to act as a small delay before going on to the next line.
L=1 is the number of the first line of text.
The COLOR Presents a light buff background which Provides a better contrast than green for reading the text on screen.

```
120 PRINT@299,"ENTER OR RETRIEVE"
```

This Provides the oPtion of using the Program for Entering text by tyPing or Retrieving the text from taPe.

```
130 A$=INKEY$:A$=""
140 A$=INKEY$:IFA$=""THEN140
150 IFA$="R"THEN720
160 IFA$<>"E"THEN140
170 CLS:PRINT@2,"LINE"L
```

INKEY$ is the command that instructs the comPuter to scan the keyboard and return any key that has been Pressed.
A$ records which key (if any) that was Pressed.
Line 130 makes A$ = INKEY$, then sets A$ to "nothing".
Line 140 rePeats A$=INKEY$ and if no key has been Pressed the comPuter stays at this line until a key IS Pressed.
Line 150 sends the Program to line 720 if R is Pressed.
If R was not Pressed and the Program goes on to line 160 then line 160 will send the comPuter back to 140 if E has not been Pressed. This means that if any key other than R or E is Pressed then the Program will go back and continue checking until either R or E is Pressed.
Line 170 after clearing the screen Prints LINE followed by the value of L, which at Present is 1.

```
180 PRINT@32,"  [graphics characters]"
190 PRINT@128,"[graphics characters]":SOUND22,1
```

These are the the TYPING GUIDE to assist in laying out the text.
Line 180 begins with two SPACES and thirty STAR and FULL STOP characters.
Line 190 has 32 characters followed by a warning SOUND.

```
200 PRINT@162,"LINE"L-1:PRINT@226,Z$(L-1)
```

Line 200 Prints the Previous line of text on the screen.

ENTER TEXT

```
210 PRINT@32,"■■":INPUTZ$(L)
220 IFZ$(L)=""THENZ$(L)=" "
```

Print at 32 two SHIFTJs. This Places the INPUT in the right Place on the screen.
This is the INPUT that will take the lines of text. The subscriPt L will be increased by 1 for each line.
Line 220 is to make sure that none of the String Variables being used for text will ever be left emPty. If you were to hit RETURN without having tyPed anything then this will Put a non-Printing character into that line. The quotes contain a SHIFTZ graPhic. This line will aPPear again later to take care of any line that is inadvertently left emPty by the EDIT subroutine.
The reason for this is that the ASCII CONVERSION subroutine that takes care of UPPER and LOWER cases, will not admit an emPty String Varaible. The Program would BREAK.

MENU

```
230 CLS:A$=INKEY$:A$=""
240 PRINT@40,"[illegible]";L:PRINT@104,"SCROLL"
250 PRINT@116,"EDIT":PRINT@168,"PRINT"
260 PRINT@232,"FILE":PRINT@296,"RETRIEVE":PRINT@360,"CONTINUE"
270 PRINT@421,"[illegible]";X:SOUND18,1
```

Clear the screen and set A$ to record the INKEY$ scan of the screen. Make sure that A$ is empty.
Lines 240 to 270 sets out the MENU on the screen. L is the TOTAL LINES used. X will be the LAST LINE EDITED. The INVERSE capital letters of each of the options is the key to be pressed to access that option.

```
280 A$=INKEY$:A$=""
290 A$=INKEY$:IFA$=""THEN290
300 IFA$="S"THEN770
310 IFA$="E"THEN460
320 IFA$="P"THEN370
330 IFA$="F"THEN670
340 IFA$="R"THEN720
350 IFA$="C"THENL=L+1:GOTO170
360 IFA$<>"C"THEN290
```

As before we are using A$ to record the keyboard scan by INKEY$.
Note that in line 350, because this will produce the TYPING GUIDES and INPUT for another line of text, L is increased by 1 to change the line number and to also change the String Variable to Z$(2).
Line 360 sends the program back to 290 should any other key be pressed.

■ You can now RUN the program to see the Menu. Of course, you cannot select an option from the Menu as the subroutines have not yet been entered. If you are using AUTO then use RUN100, and to get back to programming just use RUN. You will see that line 0 appears blank. This is because the TOKEN for AUTO has no character equivalent.

ASCII CODE CONVERSION

All characters are held in computer memory as numbers; a code of numbers known as the ASCII CODE. Printers also have all its characters stored in memory. But, while printers usually use a standard code, some computers do not. Your VeeZed holds some standard ASCII Code numbers in memory but many numbers do not conform. For instance, if you type PRINTCHR$(109) a HYPHEN is printed on the screen, but if you type LPRINTCHR$(109) a lower case m is printed on the printer. Fortunately, the computer's non-standard ASCII Code numbers fall into consecutive blocks. All the VeeZed's INVERSE letters are between ASCII numbers 192 and 223, so if we want to change an INVERSE letter to an UPPER case letter, we simply subtract 128 from every ASCII number in that block. This means that for every INVERSE letter you type an UPPER case character will be LPRINTed instead of an INVERSE letter. If we do the same to the block of normal characters to LPRINT LOWER case characters then we have the basis for a word processor. The first RULE is that your CAPITAL letters must be typed in INVERSE, while normal characters will be LOWER case.

Because your VeeZed turns COMMAS, COLONS and SEMI-COLONS into COMMANDS, they must also be entered in INVERSE, so that they too can have their ASCII numbers changed. If you type a normal COMMA in your text,the computer will print EXTRA IGNORED and the COMMA and all after will be lost. When this happens you're able to repair the damage using the EDIT facility. ANNOYING!!!

NOW FOR THE SUB-ROUTINE...

```
370 CLS:FORJ=1TOL:FORI=1TOLEN(Z$(J))
```

This line contains two loops. In computer jargon when one loop is INSIDE another it is called a NESTED loop.
The INNER loop (I) will run all its steps for every step of the OUTER loop (J).
In this case, while the OUTER loop is selecting each line of text in turn, the INNER loop will be selecting every character in each line of text in turn.
In this way we can examine every character that has been typed and change its ASCII number if necessary, before passing it on to the printer.
Seems a big task, but don't worry, your computer does it very quickly.
L is the number of text lines.
Z$(J) is each line in turn, while LEN(Z$(J)) is the length (or the number of characters) in each line.

```
380 A=ASC(RIGHT$(LEFT$(Z$(J),I),1))
```

IMPORTANT It is important that you type this line correctly. Please check that all the BRACKETS and COMMAS are exactly as shown here.

If Z$(1) is the first line of text, then LEFT$(Z$(J),1) will be the first character of that line. LEFT$(Z$(J),2) will be the first TWO characters. Now, if we apply a RIGHT$(...) to those two letters, then the RIGHT$ will be only the second character. So, if we make A the ASC, (ASCII number) of the RIGHT$ of the LEFT$ then A will become the ASCII number of each character in turn.

```
390 IFA>=64ANDA<=95THENA=A+32
400 IFA>=192ANDA<=223THENA=A-128
410 IFA>=224ANDA<=255THENA=A-192
```

Line 390 says, if A (which will be the ASCII number of each character of the text in turn) is EQUAL TO or GREATER THAN 64, AND if A is EQUAL TO or SMALLER THAN 95, then ADD 32 to A. The block of ASCII numbers from 32 to 95 are held in computer memory as ordinary letters, numbers and punctuation marks. By "ordinary" we mean NOT INVERSE. If they were LPRINTed the letters would be CAPITAL letters. We had decided that VeeZed INVERSE letters would be UPPER case. These ordinary letters are to be used for LOWER case. By increasing the value of A by 32 in this block then they will be the ASCII numbers accepted by the printer as LOWER case letters. Simple, eh?

Line 400 deals with the block of VeeZeds INVERSE characters, changing them by subtracting 128 so they will LPRINT UPPER case. And line 410 does a similar change to INVERSE numbers and punctuations so that these can be entered in INVERSE or as ordinary characters. The main point of this line is that this block includes a COMMA, a COLON and a SEMI-COLON. These three characters, when entered in text lines, causes the VeeZed to produce the error message EXTRA IGNORED, so the only way round the problem was to only use them in INVERSE. This line does that.

```
420 REM
430 REM
```

These lines are for sending Printer Control Codes to change the status of a printer. Some printers can be made to print *ITALICS*, in SUPERSCRIPT and SUBSCRIPT, or even to UNDERLINE. Obviously, we cannot put in Printer Controls unless we know what printer is to be used. Later we will give details of how to use these lines for the GP-100, to make that printer change to **EXPANDED** printing.

```
440 LPRINTCHR$(A);:NEXT
```

LPRINTCHR$(A); because it is followed by a SEMI-COLON does NOT LPRINT. Instead, it loads the value of A into the printer's memory, which is called the BUFFER. NEXT, of course, sends the INNER loop (I) back for another A.

```
450 LPRINT:PRINTZ$(J):NEXT:GOTO230
```

This LPRINT does LPRINT. It releases all the characters stored in the BUFFER of the printer so that a whole line is LPRINTed. NEXT continues the OUTER loop (J) which goes back for another line of text.
PRINTZ$(J) comes before NEXT so that the line, as typed, will be printed on the screen, keeping you informed of the progress of this sub-routine.
Finally, GOTO230 sends you back to the MENU.

EDITING SUB-ROUTINE

This sub-routine may appear complex. Don't worry if my explanation leaves you starry eyed. Instead of trying to explain it in detail, I will just give a general indication of what it is doing.

```
460 CLS:PRINT@40,"EDIT":SOUND14,3:
PRINT@45,"LINE";:INPUTC:X=C
470 IFC>LTHEN460
```

NOTE Underlined characters are to be entered in INVERSE.

Line 460 after clearing the screen, prints the word EDIT after a short SOUND, and adds the word LINE. The INPUT puts a question mark after LINE, making it an invitation to enter the line number you wish to Edit. X is made equal to C. X will be used in line 270 of the Menu to show which line was last edited. Line 470 says if the line number entered is MORE THAN the the number of lines in the text, (L), then go back to 460 and try again.

```
480 PRINT@0,"LINE"C
490 PRINT@32,"      ZZZZZZZZZZZZZZZZZZZZZZZZZZ"
500 PRINT@192,Z$(C)
510 PRINT@128,"ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ"
520 PRINT@32,"DELETE":INPUTR$ D$
530 B=64:FORD=0TO63:PRINT@B," ":B=B+1:NEXT
540 IFD$=""THEN230
```

Line 480 prints the line number at the top of the screen.
Line 490 prints lines similar to the Typing Guides. There are six spaces, (where the word DELETE is going), followed by 26 SHIFTZ or SHIFTJ, if you prefer, Graphics. Then line 500 will print the line requested for editing. Line 510 prints 32 SHIFT Zs (or SHIFTJs) and 520 adds the word DELETE and puts an INPUT question mark between the Guide Lines. This is where you type the letter or letters, word or words that have to be DELETED.
Line 530 wipes out whatever you've entered in preparation for you to type the correction. Line 540 is there to return you to the menu if you have entered nothing at all.

```
550 Q=LEN(D$):S=LEN(Z$(C))
560 SOUND30,1:PRINT@32," ENTER":INPUTR$
```

Q becomes the length of your DELETE, and S becomes the length of the original line of text. Q and S will be used shortly.
Line 560 prints ENTER over the top of DELETE. Because ENTER is one letter short we have added an INVERSE SPACE in before the letter E. The INPUT puts a question mark inside the Guides.
This time you type in the letter or letters, word or words to be ENTERed in place of those to be DELETEd.
It is important that the letter(s) or word(s) to be deleted are EXACTLY as originally typed or the computer will not be able to recognise them.
Using the LEN(...) information the computer is going to look for your DELETE characters. If it cannot find them it will let you know.

```
570 FORZ=1TOS
580 IFMID$(Z$(C),Z,Q)=D$THEN620
590 NEXT
600 PRINT@97,"INCORRECT"
610 SOUND4,9:GOTO230
```

The loop is for finding the character(s) to be deleted. If the computer finds them it will send the program to line 620, but if it does not find them it prints the word INCORRECT and after blowing you a *raspberry* puts you back to the menu.

```
620 E=Z-1+LEN(D$)
630 N$=LEFT$(Z$(C),Z-1)+R$+RIGHT$(Z$(C),LEN(Z$(C))-E)
640 Z$(C)=N$
```

This is another place where you should be careful to get all those COMMAS and BRACKETS correct. This is where the computer uses a temporary String Variable (N$) to build the new line of text by replacing your DELETE with your ENTER.

```
650 FORD=192TO256:PRINT@D,CHR$(32):NEXT
660 PRINT@192,Z$(C):FORT=1TO1000:NEXT:GOTO230
```

Line 650 erases the original line of text from the screen. This method is just an alternative to the method used in line 530. CHR$(32) is the same as " " (just a SPACE).
Line 660 prints the new EDITED line of text with a FOR/TO loop to provide sufficient time for you to see if it is correct before going back to the menu.

All that remains now is the sub-routines for FILING TO TAPE, RETRIEVING FROM TAPE and SCROLLING the text up the screen.

IT MIGHT BE A GOOD IDEA NOW TO TEST RUN, DEBUG AND CSAVE.

TAPE FILE

```
670 CLS:PRINT@68,"TAPE FILE":SOUND16,2:PRINT@235,"RECORD"
680 SOUND22,9:CLS
690 FORJ=1TOL
700 PRINT#"",Z$(J):PRINTZ$(J):SOUND30,1:NEXT
710 CLS:PRINT@235,"STOP TAPE":SOUND24,9:CLS:GOTO230
```

Line 670. After clearing the screen the words TAPE FILE are printed and then followed by RECORD. A couple of SHIFTJs either side of RECORD can help to dress up the screen display.
Line 680. The long sound provides the opportunity for the tape to be started on RECORD.
Line 690. Here we begin the loop for recording the text from line 1 to line L, which is the total number of lines entered.
Line 700. This is the command for recording to tape.
The quotes normally contain a FILENAME, but as we cannot use a String Variable to change the Filename each time, we have left the quotes empty. As we have done the same in the RETRIEVE sub routine, it means that this program cannot be used to recover data that has been recorded with a Filename, unless the Filename is added.
As each line of text is recorded it is printed on the screen with a short SOUND.
Line 710. When the recording is completed STOP TAPE appears. A long sound acts as a warning that the recording is finished. Then back to the menu.

RETRIEVE FROM TAPE.

```
720 CLS:PRINT@226,"RETRIEVE":SOUND22,6:PRINT@290,LINES";:
    INPUTL
730 CLS:SOUND24,5
740 FORJ=1TOL:INPUT#"",Z$(J):PRINTZ$(J):NEXT
750 PRINT@490,"STOP TAPE":SOUND26,9
760 GOSUB790:GOTO230
```

Line 720. Clear the screen, print the title RETRIEVE and after a SOUND print the word LINES. The INPUT puts a question mark after the word LINES inviting you to enter the number of lines to be retrieved.
Line 730 clears the screen and the SOUND tells you it is time to start the tape on PLAY.
Line 740. Here again is the loop which runs from line one to L which is the total number of lines of text to be retrieved. Each line appears on the screen as it is retrieved.
Line 750. STOP TAPE is printed on the screen with a long SOUND to let you know that the Retrieval is complete.
Line 760. GOSUB790 picks up a DELAY loop which is part of the SCROLL sub-routine then returns you to the menu.

SCROLL

```
770 CLS:FORJ=1TOL:PRINTJ:PRINTZ$(J):GOSUB790:NEXT:GOSUB790
780 GOTO230
790 FORT=1TO1600:NEXT:SOUND20,1:RETURN
```

SCROLL is to have your text printed on the screen line-by-line to give you the opportunity of checking the text for errors. The line number is printed just before the line itself so that you can make a note of which lines need to be corrected. The SCROLLING speed needs to be suited to your own reading speed. This is done by changing line 790. To make the SCROLL run more slowly just change 1600 to a larger number.
Line 770 is the now familiar loop running from text line 1 to L (the total number of lines). Firstly it prints on the screen the line number (J), then the line followed by GOSUB790 which is the DELAY loop. Another GOSUB790 follows NEXT to give you time to read the last line of text before getting back to the menu (line 780).

That is the end of the **MICRO TYPE** program except for making it suitable for whatever Printer or Plotter you will be using.

PRINTERS and PLOTTERS.

MICRO MAGIC is presented here in its most simple form. There is no provision for Printer Control Codes. These codes will be discussed in the Studio Ad Lib pages. However we will now give you details on how to use this program on a GP-100 Printer and on a VeeZed PP40 Plotter and a TOYO PP-40 Plotter.

GP-100 Graphic Printer

This printer is especially suited to both the VZ-200 and VZ-300 as it simulates both MODE(0) and MODE(1) graphics. The ASCII Code is standard in respect of all text characters. The characters for graphics use the same ASCII coding as the VeeZed, which means you can LLIST any program whether it contains graphics or not. This cannot be done on other printers as the graphics in a program would print garbage. A disadvantage with the GP-100 is that the graphic characters take up more space than the text characters so when a COPY of a screen containing both test and graphics is done, (called a SCREEN DUMP), then the print-out is a bit of a mess. Another problem is that the text characters do not have DESCENDERS. They are that part of letters such as g, j, p, q and y that descend below the print line. Apart from making your text look strange, (a lower case p can look like a capital P), it also overprints an UNDERLINE on top of the characters being underlined.

The GP-100 does print **EXPANDED** characters. To use this facility in MICRO TYPE then here are the two lines to be added to the program. We already reserved line 430 with a REM.

```
420
430 IFA=126THENA=0:GOSUB800
800 LPRINTCHR$(14);:RETURN
```

The UP-POINTING ARROW (SHIFTN) is to be used to put the GP-100 Printer into EXPANDED mode. When you enter a line of text you put a SHIFTN immediately before any letter or word(s) that you want printed in large-size print. The SEMI-COLON in line 800 is to prevent a linefeed and carriage return. Where line 420 says A=0 this is to give A a NUL. A NUL makes the ARROW disappear! It will not LPRINT and will not leave a SPACE, but the GOSUB will send CHR$(14) to the printer to change to Expanded mode. Brilliant!

All we need now is a similar method of cancelling the expanded mode to continue printing in normal-size characters.

```
430 IFA=124THENA=0:GOSUB810
810 LPRINTCHR$(15);:RETURN
```

With these lines in place we can use SHIFTM, (the BACKSLANT), to CANCEL the expanded mode. To put a letter or a word or a phrase into EXPANDED print you must type a SHIFTN just before and a SHIFTM just after the word. Try it.

Incidentally, the ASCIIs for SHIFTN and SHIFTM are 94 and 92. We have used 126 and 124 because 94 and 92 were increased by 32 in line 390. We could have squeezed the two new lines in before line 390, in which case the proper ASCIIs (94 and 92) would have been used. In Studio Ad Lib we will give you a program that will demonstrate the problem with the UNDERLINE.

To use MICRO TYPE with the PP40 PLOTTER it is necessary to either change the plotter to 80-characters per line or change the Typing Guides in the program to accommodate only 40. Page 7 of the plotter's Instruction Manual shows how to change the switch to either 40 or 80 columns, which is the same as 40 Or 80 characters per line. A plate has to be removed from under the plotter and switch no.2 must be set to the ON position for 40 columns, and OFF for 80 columns. Make sure the power is off when you operate the switch. The program will work on 40 just as written, but the plotter will put in a CARRIAGE RETURN and LINE FEED whenever the printing reaches the fence.

The TOYO TP-40 Printer/Plotter is similar to the Dick Smith PP40, except that the switches are readily accessable on the bottom. Page 7 of the Operation Manual shows how to change the switches. To change to 80 characters per line, switch 1 has to be pushed to the LEFT, and to select 40 characters per line it must be pushed to the RIGHT. Make sure the power is off when changing this switch.

Dick Smith PP40 PLOTTER.

TOYO TP-40 PRINTER/PLOTTER.

In Studio Ad Lib we will be giving details of how Printer Control Codes are sent from the Vee2ed to printers. If you do not have a GP-100 you may be able adapt the MICRO TYPE program to suit your particular printer. If you have any difficulty just write, telling me what printer you have and in what form your Instruction Book says the Control Codes take.

ASCII CODES

DEC.	HEX.	CHAR.	DEC.	HEX.	CHAR.	DEC.	HEX.	CHAR.
000	00	NUL	044	2C	,	087	57	W
001	01	SOH	045	2D	-	088	58	X
002	01	STX	046	2E	.	089	59	Y
003	03	ETX	047	2F	/	090	5A	Z
004	04	EOT	048	30	0	091	5B	[
005	05	ENQ	049	31	1	092	5C	\
006	06	ACK	050	32	2	093	5D	]
007	07	BEL	051	33	3	094	5E	^
008	08	BS	052	34	4	095	5F	<
009	09	HT	053	35	5	096	60	`
010	0A	LF	054	36	6	097	61	a
011	0B	VT	055	37	7	098	62	b
012	0C	FF	056	38	8	099	63	c
013	0D	CR	057	39	9	100	64	d
014	0E	SI	058	3A	:	101	65	e
016	10	DLE	059	3B	;	102	66	f
017	11	DC1	060	3C	<	103	67	g
018	12	DC2	061	3D	=	104	68	h
019	13	DC3	062	3E	>	105	69	i
020	14	DC4	063	3F	?	106	6A	j
021	15	NAK	064	40	@	107	6B	k
022	16	SYN	065	41	A	108	6C	l
023	17	ETB	066	42	B	109	6D	m
024	18	CAN	067	43	C	110	6E	n
025	19	EM	068	44	D	111	6F	o
026	1A	SUB	069	45	E	112	70	p
027	1B	ESC	070	46	F	113	71	q
028	1C	FS	071	47	G	114	72	r
029	1D	GS	072	48	H	115	73	s
030	1E	RS	073	49	I	116	74	t
031	1F	US	074	4A	J	117	75	u
032	20	SPACE	075	4B	K	118	76	v
033	21	!	076	4C	L	119	77	w
034	22	"	077	4D	M	120	78	x
035	23	#	078	4E	N	121	79	y
036	24	$	079	4F	O	122	7A	z
037	25	%	080	50	P	123	7B	{
038	26	&	081	51	Q	124	7C	|
039	27	'	082	52	R	125	7D	}
040	28	(	083	53	S	126	7E	~
041	29	)	084	54	T	127	7F	DELETE (Rubout)
042	2A	*	085	55	U			
043	2B	+	086	56	V			

STANDARD ASCII ABBREVIATIONS

ACK = Acknowledge
BELL = Bell
BS = Backspace
CAN = Cancel
CR = Carriage Return
DC1 = Direct Control 1
DC2 = Direct Control 2
DC3 = Direct Control 3
DC4 = Direct Control 4
DLE = Data Link Escape
EM = End of Medium
ENQ = Enquiry
EOT = End of transmission
ESC = Escape
ETB = End Transmission Block
ETX = End Text
FF = Form Feed
FS = Form Separator
GS = Group Separator
HT = Horizontal Tab
LF = Line Feed
NAK = Negative Acknowledge
NUL = Null
RS = Record Separator
SI = Shift In
SO = Shift Out
SOH = Start of Heading
STX = Start Text
SUB = Substitute
SYN = Synchronous Idle
US = Unit Separator
VT = Vertical Tab